

AD-A216 789

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

(2)

REPORT DOCUMENTATION PAGE				
1. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		10. RESTRICTIVE MARKINGS		
2. SECURITY CLASSIFICATION UNCLASSIFIED		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE		5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR-89-1825		
4. PERFORMING ORGANIZATION REPORT NUMBER D-65		7a. NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research		
6a. NAME OF PERFORMING ORGANIZATION University of Massachusetts Computer & Info. Sci. Dept.		7b. ADDRESS (City, State and ZIP Code) Bolling AFB DC 20332-6448		
6c. ADDRESS (City, State and ZIP Code) Amherst, MA 01003		7d. ADDRESS (City, State and ZIP Code) Bolling AFB DC 20332-6448		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-87-030 FQ8671-8700280 (purchase request no.)		
8b. OFFICE SYMBOL NL		10. SOURCE OF FUNDING NOS.		
8c. ADDRESS (City, State and ZIP Code) Bolling AFB DC 20332-6448		PROGRAM ELEMENT NO. 61102F		
11. TITLE (Include Security Classification) Cooperative Interaction of ...Processing Units		PROJECT NO. 2312		
12. PERSONAL AUTHOR(S) Andrew G. Barto		TASK NO. A1		
13a. TYPE OF REPORT Final Technical		WORK UNIT NO. 5		
13b. TIME COVERED FROM 861001 TO 890930		14. DATE OF REPORT (Yr., Mo., Day) 891130		
15. PAGE COUNT 35		16. SUPPLEMENTARY NOTATION Recomm.		
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB. GR.	Adaptive networks Learning Neural computing Stochastic learning automata Connectionist systems Cooperative computing	
19. ABSTRACT (Continue on reverse if necessary and identify by block number)			This report describes progress made in the development of connectionist learning methods permitting networks to learn when they cannot be provided with training information of the high quality required by supervised-learning methods. These methods can permit the application of adaptive connectionist networks to tasks involving complex dynamical behavior and high degrees of uncertainty. A method for training layered networks to perform nonlinear pattern recognition and associative memory tasks was refined. The neuron-like units making up these networks learn on the basis of feedback that evaluates behavior but does not specify desired output or directly provide error information. We report how this method is related to gradient-following methods, how its learning rate can be improved, and argue that this method is biologically plausible. A generalized theory of supervised learning was developed, in which training information comes in the form of constraints instead of specifications of desired network outputs.	
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input checked="" type="checkbox"/> DTIC USERS <input type="checkbox"/>			21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL William O. Berry			22b. TELEPHONE NUMBER (Include Area Code) (202) 767-5021	
22c. OFFICE SYMBOL NL				

19. Abstract (cont.)

This approach was illustrated by using it to train a simulated multi-jointed manipulator to perform sequences of reaching tasks. A theoretical perspective was developed for addressing issues such as what happens when network learning algorithms are scaled up to larger tasks. A very general formulation of the network learning task was shown to possess no efficient general solution independently of the learning algorithm used. We report progress on methods for structuring networks and their training in directions that may allow these computational limitations to be overcome. Additional progress was made in the development of reinforcement learning methods for control of dynamical systems.

FINAL TECHNICAL REPORT: AFOSR-87-0030
COOPERATIVE INTERACTION OF SELF-INTERESTED
NEURON-LIKE PROCESSING UNITS
Principal Investigator: Andrew G. Barto

Summary—Progress was made in the development of connectionist learning methods permitting networks to learn when they cannot be provided with training information of the high quality required by supervised-learning methods. These methods can permit the application of adaptive connectionist networks to tasks involving complex dynamical behavior and high degrees of uncertainty. A method for training layered networks to perform nonlinear pattern recognition and associative memory tasks was refined. The neuron-like units making up these networks learn on the basis of feedback that evaluates behavior but does not specify desired output or directly provide error information. It was shown how this method is related to gradient-following methods, how its learning rate can be improved, and it was argued that this method is biologically plausible. A generalized theory of supervised learning was developed, in which training information comes in the form of constraints instead of specifications of desired network outputs. This approach was illustrated by using it to train a simulated multi-jointed manipulator to perform sequences of reaching tasks. A theoretical perspective was developed for addressing issues such as what happens when network learning algorithms are scaled up to larger tasks. A very general formulation of the network learning task was shown to possess no efficient general solution independently of the learning algorithm used. Progress was made on methods for structuring networks and their training in directions that may allow these computational limitations to be overcome. Additional progress was made in the development of reinforcement learning methods for control of dynamical systems.



Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and for special
A-1	

1 Research Accomplishments

Following is the abstract of the research proposal that led to funding of the research being reported here. It states the research objectives.

The aim of the proposed research is to extend experience with a particular approach to learning in networks of neuron-like adaptive elements. The long-term goal of this research is the development of massively parallel adaptive systems that incorporate principles of operation suggested by nervous systems. It is an alternative to the knowledge-based approach of conventional Artificial Intelligence. The chief characteristic of our approach is that each network component is a self-interested agent that attempts to learn, via a reinforcement learning process, how to obtain its most highly preferred inputs. These components implement a novel learning rule previously developed by us that causes them to learn to enter into cooperative interactions with one another for mutual benefit. A significant consequence of this type of interaction is that layered networks of these elements can learn to implement nonlinear associative mappings by constructing the necessary representations. This method of learning nonlinear associative mappings is therefore one of several recently developed by various research groups that promises to greatly extend the power of adaptive networks. We propose to exploit the capabilities of this method in control tasks related to the following problems: 1) the development of coordinative structures in motor control, and 2) the learning of strategies and representations for guiding movement in space. In the context of these domains, we propose to investigate how well these algorithms scale up to larger networks, how their performance can be improved, and how their novel capabilities can extend current abilities to control complex systems. The proposed methodology relies on computer simulation and mathematical analysis.

We have made substantial progress in the following areas: 1) the refinement of reinforcement learning methods for nonlinear pattern recognition and associative memory, 2) the development of adaptive network methods applicable to problems in motor control and robotics, 3) the development of a theoretical perspective on scaling up network learning algorithms, 4) the investigation of methods for accelerating convergence of learning methods by adaptively altering learning rate parameters, 5) the development of a modular network architecture and learning method, and 6) the refinement of reinforcement learning methods for control of dynamical systems. I discuss each of these areas of progress below. Several additional topics are discussed which were not as well developed as those mentioned above at the end of the funding period. Most of the research conducted under this grant has been described in detail in published technical reports, conference papers, and journal articles. In the sections that follow, the topics on which we have published

are treated with less detail than those on which we have not yet published, and references to the appropriate published material are provided.

Although we believe that all of this progress has been significant, we believe that some of the research is truly outstanding, having already made a considerable impact on the field of connectionist computing. Specifically, the theoretical results achieved by J. S. Judd, supported as a graduate student by the grant, on the complexity of network learning is having considerable influence, as is the research of M. I. Jordan, supported by the grant as a post-doctoral researcher, on supervised learning for systems with excess degrees of freedom.

2 Refinement of Reinforcement Learning Methods

Substantial progress was made in refining and exploring the utility of neuron-like units that attempt to maximize an evaluation, or reinforcement, signal. Unlike most of the neuron-like adaptive units being studied by others, such units do not require direct specification of target, or desired, outputs. In this section, we focus on reinforcement learning methods as applied to classification tasks and static decision tasks. In Section 7 we discuss more recent progress made in understanding reinforcement learning as an approach to the problem of controlling dynamical systems.

2.1 The Associative Reward-Penalty (A_{R-P}) Unit

An A_{R-P} unit is a neuron-like unit having a stochastic binary output and a learning rule for adjusting its connection weights so as to maximize the probability with which it receives an evaluation or reinforcement signal indicating "reward" or "success" (Figure 2.1). It is one embodiment of the idea of a "heterostat" put forward by Klopff [22, 23]: a neuron-like unit which tries maximize some local form of "pleasure". The A_{R-P} unit is the most successful of our efforts to develop Klopff's idea into a concrete implementation. Barto and Anandan [4] described the learning rule and proved that it maximizes success probability under certain conditions. In previous research, we found that layered networks of A_{R-P} units could reliably learn to solve nonlinear pattern classification and associative memory tasks [2, 1]. After we developed the A_{R-P} unit and applied it to the problem of learning of nonlinear mappings by layered networks, the error back-propagation algorithm was popularized by Rumelhart, Hinton, and Williams [30]. The success of this latter algorithm, and the publicity surrounding it, has been partly responsible for the enormous increase in interest in artificial neural networks. Obviously, therefore, we were interested in investigating the relationship between the A_{R-P} and the error back-propagation methods and to compare their performances. Previous comparative simulations [1] showed that back-propagation is considerably faster than the A_{R-P} method in some simple tasks, but we felt that the A_{R-P} method might still have some advantages, especially if we could develop some of the obvious ways of speeding it

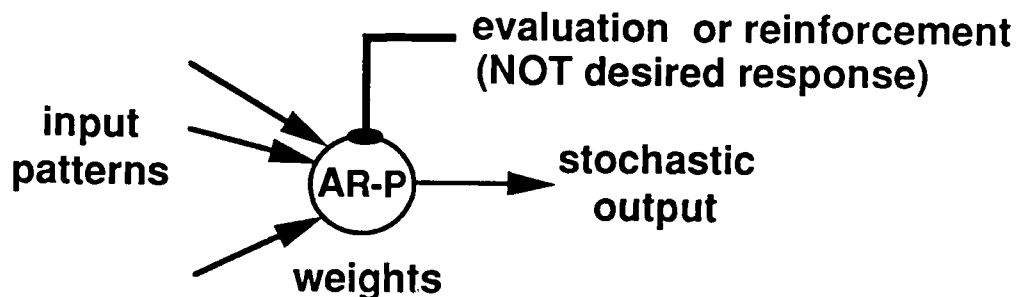


Figure 1: An Associative Reward-Penalty (A_{R-P}) Unit.

up.

We wanted to take advantage of the theoretical results of Williams [38, 39] showing that the weight vector of an A_{R-P} network follows a relevant performance gradient in a statistical sense; that is, A_{R-P} networks perform a stochastic form of gradient descent. Based on these results, we developed a way to apply the A_{R-P} algorithm to supervised learning tasks so that the network does something as close as possible to what back-propagation does, but does it without requiring the complex back-propagation process. We also introduced a method, which we called "batching," for increasing the learning efficiency of A_{R-P} networks. In this method we essentially let the A_{R-P} units in the network generate several actions while network input is held constant at one of the training patterns. For each of these actions the network receives a reinforcement value. Weights are then updated according to the resultant weight change computed over this period of constant input. We applied this method to the task of learning to detect symmetry axes in binary patterns on a four-by-four grid. In one version of the task, the network has three output units, and must categorize the input as having either horizontal, vertical, or diagonal symmetry (only one of the two possible diagonal axes is used). We also studied a simpler task with a single output unit, in which the network must discriminate between horizontal or non-horizontal symmetry. For either version of the task, our networks had sixteen input units, corresponding to the four-by-four grid, and twelve hidden units. There was full connectivity between layers, yielding a total of 243 modifiable weights and biases in the case with three output units, and 217 modifiable weights and biases in the case with one output unit.

The results showed that batching can speed up learning by A_{R-P} networks, but back-propagation is still faster. Nevertheless, the A_{R-P} method may have some advantages over back-propagation: 1) it is much more plausible from a biological perspective, 2) it may be easier to implement in hardware, and 3) it is applicable to tasks to which back-propagation is not, namely, tasks in which desired responses are not known for a set of training instances. Barto and Jordan [6] described these results in detail. This paper also introduced a version of the A_{R-P} algorithm, called the "S-model A_{R-P} ", that works for real-valued reinforcement signals instead of the binary ones to which the original A_{R-P} method was restricted.

Due to the superiority of the error back-propagation method over A_{R-P} networks as a practical method for training layered networks in supervised-learning tasks, we shifted attention to tasks in which the training information required for supervised learning is absent. In these tasks, called associative reinforcement learning tasks, the target responses of a network's output units are not known, but the consequences of the network's output patterns on an unknown process can be evaluated by a critic which at each step supplies the network with an evaluation, or reinforcement, signal. Figure 2 shows how a network can be applied to such a task. The critic in the figure is shown as supplying signals to a reinforcement pathway for each unit in the network, but the important point is that learning occurs even when all these pathways always transmit the *same signal* to their target units, i.e., when a single evaluation signal is effectively broadcast to all the units. If the critic is able to differentiate this global evaluation by sending evaluations specialized for individual units, learning occurs more quickly. Thus, although A_{R-P} units can take advantage of individualized evaluation signals, they are also able to learn as members of a "team" seeking collective behavior that maximizes reinforcement [2].

In associative reinforcement learning tasks such as shown in Figure 2, the reinforcement learning abilities of A_{R-P} units are needed not just for the hidden units but for the output units as well. One task of this type with which we experimented has been called a "decentralized team decision problem" (e.g. ref. [11]). We performed some computational experiments using A_{R-P} units as the decision makers in a simple example of a decentralized team decision problem. In this task, there were two decision makers each making a decision based on different but correlated information about some underlying uncertainty. The outcome of the decisions depended on the coordinated actions of the units. The units had to learn how to maximize a payoff through repeated trials. We found that A_{R-P} units were consistently able to learn how to solve this task. This is a learning task in which uncertainty plays an intrinsic role, and to which supervised-learning methods are not directly applicable. This task is described in a book chapter by Barto [3], which summarizes a range of results we have achieved involving the collective behavior of reinforcement-learning units. It also develops the hypothesis that there may be a close relationship between neuronal learning rules and chemotactic strategies of freely-living unicellular organisms.

Another aspect of our research on the A_{R-P} unit has been our continuing attempts to prove a stronger version of the A_{R-P} convergence theorem proved by Barto and Anandan

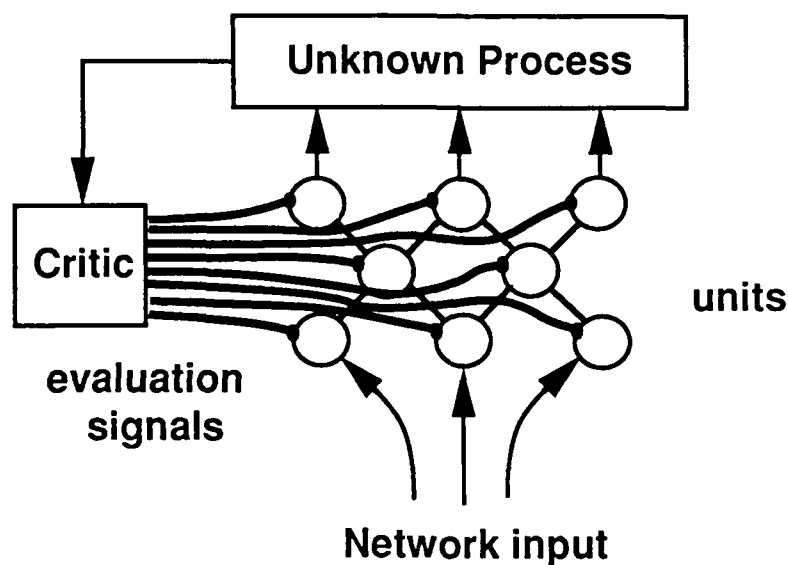


Figure 2: An $AR-P$ network in an associative reinforcement learning task. Effective learning occurs even when all the evaluation pathways transmit identical signals.

[4]. According to this theorem, an $AR-P$ unit successfully maximizes success probability under rather general conditions, but the theorem requires one condition which is quite stringent. Specifically, although the theorem allows success probabilities to depend on $AR-P$ unit actions in the most general probabilistic way (that is, for a given input pattern, *any* process whatsoever can determine how success or failure signals depend on unit activity), the theorem requires that the input patterns to the unit be linearly independent. This is the same condition ensuring that a unit using the Widrow-Hoff, or Least Mean Square, learning rule in a supervised task can solve that task exactly. The theorem sheds no light on what happens when the input patterns are not linearly independent, which is the usual situation in pattern classification tasks. We have been trying to prove a result which does not require the assumption of linear independence.

Unfortunately, to remove this assumption requires a proof technique different from the one Barto and Anandan employed. Consequently, we arranged to consult with Dr. P. S. Sastry of the Indian Institute of Science, Bangalore, India, an expert in applying theories of stochastic convergence to learning problems, who was visiting this country. We worked on several approaches to proving a stronger convergence theorem, and were able to arrive at results for a special case of the $AR-P$ algorithm. However, our work with Sastry indicated that the desired general result for the the $AR-P$ algorithm is not going to be easy to prove because it seems to involve a very complex stochastic process. We did not abandon our goal of proving a stronger $AR-P$ convergence theorem, but we placed a low priority on it.

2.2 Real-Valued Stochastic Reinforcement Learning

A limitation of the A_{R-P} unit for applications to motor control problems is that it is a binary unit, that is, it has just two actions. In control tasks it is usually important to be able to provide control signals with continuous values. It turns out to be nontrivial to extend the reinforcement learning principles used by the A_{R-P} unit to the case of real-valued outputs. Basically, the difficulty lies in that fact that the probability mass or density function over the set of unit actions has to be adjusted with experience by adjusting certain parameters. In the case of just two actions, adjustment over the complete set of all possible action probabilities is possible by adjusting a single parameter, which in the case of the A_{R-P} unit, is the unit's weighted sum. This is because a probability mass function for two actions has a single degree-of-freedom (i.e., given that the probability of action 1 is p , then the probability of action 2 must be $1 - p$). However, for three or more actions, you need more than one parameter.

We developed a reinforcement learning algorithm for a neuron-like unit having real-valued outputs. We called such a unit an SRV (Stochastic Real-Valued) unit. An SRV unit's output values are generated by sampling from a Gaussian distribution. The mean of the distribution is the weighted sum of inputs using one set of weights, and the variance of the distribution is determined from a weighted sum of inputs using another set of weights. The learning rule adjusts both sets of weights as a result of reinforcement feedback so that the mean moves toward the optimal action for each input pattern as the variance decreases. The performance of SRV units was studied on simple associative reinforcement learning tasks (for example, AND and XOR), with good results. We also applied SRV units to a simulated motor control task as described in Section 3 below.

We also studied the convergence properties of SRV units from a theoretical perspective. We used Martingale theory to analyze the behavior of simplified versions of the SRV algorithm, and have been able to prove convergence of the weights under certain conditions. The proof handles the associative aspects of these units in a manner similar to the A_{R-P} convergence proof of Barto and Anandan [4]. We are in the process of preparing this work for submission to a journal. The SRV research was conducted by V. Gullipalli, a research assistant funded by this grant. A technical report was published describing the SRV learning algorithm and the simulation results [10]. A version of this report is in review for the journal *Neural Networks*.

3 Motor Control and Robotics

Considerable progress was made on developing connectionist methods for handling important problems involving the control of systems with many degrees-of-freedom. In this section we focus on the research specifically involving tasks in motor control and robotics. However, the methods described are also applicable to other types of tasks.

3.1 Sequence Learning for Systems with Excess Degrees of Freedom

This research was a continuation of the Ph.D. research of M. I. Jordan, who was a post-doctoral researcher funded by this grant. In his Ph.D. dissertation [14], completed in 1985 under the direction of David Rumelhart and Donald Norman at UCSD, Jordan developed networks capable of learning sequences of patterns. Jordan's approach differs in several ways from previously studied network methods for learning sequences. First, his networks incorporate a kind of central pattern generator. Through internal recurrent connections, his networks use internal state information to generate temporal patterns without the usual form of pattern-to-next-pattern chaining. Second, Jordan used a training method in which the network's output units are given *constraints* on their actions instead of explicit instruction as to exactly what those actions should be. A consequence of this training method is that the degrees-of-freedom of an output pattern that are left unspecified by the external trainer become determined by the *temporal context* of the pattern. In other words, the specific way the required constraints are met for an output pattern at a specific time is determined by what the constraints were in the past and what they will be in the future. The result is a smooth, efficient sequence of actions satisfying the given constraints.

A natural extension of this approach is to apply it to positioning tasks for multi-jointed robot manipulators. How can one learn what joint angles produce desired end-effector positions specified in spatial coordinates (or other coordinates, such as eye-position coordinates, that are not joint angles)? A critical aspect of this inverse kinematic problem is how to choose from a usually infinite set of joint angles that yield the same desired end-effector position. This occurs when the forward kinematic transformation implemented by the manipulator does not have a unique inverse due to excess degrees-of-freedom in its structure. Resolving this kind of redundancy is an important problem in robotics, and several approaches to it have been studied. However, the approach based on Jordan's work differs from conventional approaches and represents an important contribution to the field. Using the network architecture for generating sequences of patterns, it is possible for a system to learn sequences of positioning tasks in such a way that the problem of excess degrees-of-freedom is resolved according to the temporal context of each positioning task in the sequence. How the system learns to select, from an infinite number of possibilities, a joint configuration that achieves any given target position in space is determined by the target positions that precede and follow the given target position. Configurations are selected which tend to minimize the amount of movement that must be made in moving from position to position.

To accomplish this, Jordan combined his approach to learning sequences with a method for using the error back-propagation algorithm to learn an inverse kinematic transformation. The strategy is to learn a forward kinematic transformation—a model of the robot arm—by a layered network in a “babbling” phase during which random joint angles are associated with the resulting end-effector spatial positions. Then to learn how to reach for specified spatial targets, the spatial error is back-propagated through the

network that forms the arm model to transform it into a vector of joint-angle errors used for training another layered network. Input to this second network comes from Jordan's recurrent sequence generating network, and the whole system is instructed to execute a *sequence* of reaching tasks. He showed that the network can take advantage of the arm's redundancy to find sequences of arm configurations in which the solutions at each point in time depend on the solutions found at nearby points in time, so that the redundancy is used to allow actions to overlap efficiently. This approach was demonstrated with a simulated six degree-of-freedom manipulator in a two degree-of-freedom world, as well as with a simulated manipulator with two fingers. These manipulators were trained to perform sequences of positioning tasks. These results were described in a technical report [15] and a book chapter [16].

In ref. [15] Jordan develops the theory underlying the approach as a theory of "generalized supervised learning." As a result of this theoretical perspective, it is possible to see how the approach can be applied to a variety of problems involving systems with excess degrees-of-freedom. These publications have been widely distributed, and the research they describe is exerting considerable influence on the field. Jordan left the project in January 1988 to take a position as Assistant Professor in the Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, where he is continuing this line of research.

3.2 Inverse Kinematics via Reinforcement Learning

We also investigated the use of reinforcement learning methods for problems in robotics such as the inverse kinematic problem. The purpose of these simulations was to investigate the utility of reinforcement learning as an alternative to Jordan's method described above in tasks involving excess degrees-of-freedom. Whereas the approach pursued by Jordan requires the learning of a forward model of the robot arm in order to translate spatial errors into joint errors, the approach using reinforcement learning dispenses with the necessity of such a model. We experimented with a network consisting of three SRV units and 16 hidden "back-prop" units (Figure 3) to the problem of learning a positioning task with a simulated robot arm with three degrees of freedom (Figure 4). The task was set up so that there were excess degrees-of-freedom. The planar arm has two joints and its base can move along the top axis shown in Figure 4. A positioning task was specified by selecting a target position, x_d , which was considered attained if the end of the arm stopped anywhere on the vertical line at x_d . Clearly this can be accomplished with an infinite number of different arm configurations. The evaluation signal which was broadcast to the output SRV units provided a scalar measure of the spatial distance of the end of the arm from the target line at x_d . The hidden units were trained via back-propagation of the estimated evaluation gradient. Figure 4 shows one sequence of arm positions generated by the network shown in Figure 3 after learning. Results suggested that SRV units can be useful in learning tasks involving excess degrees-of-freedom [10].

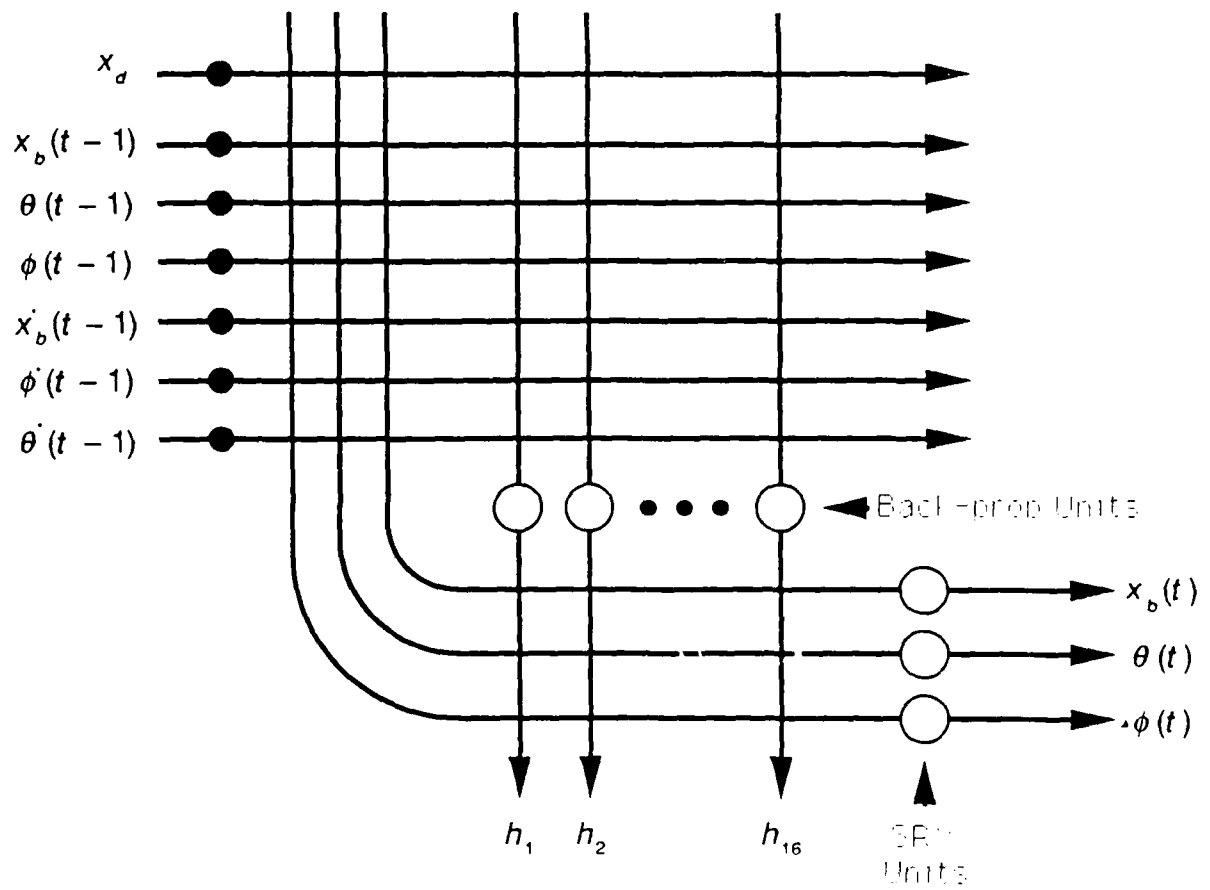


Figure 3: A network for a simulated arm positioning task consisting of three SRV output units and 16 hidden units.

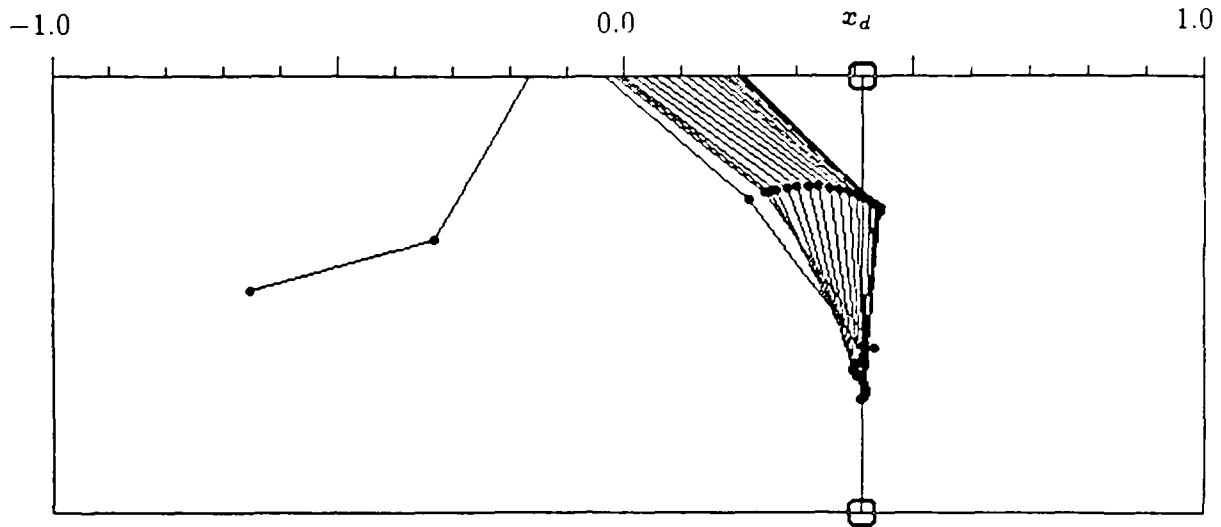


Figure 4: The three degree-of-freedom planar arm used to study SRV units in positioning tasks. A target position is reached when the end of the arm stops anywhere along the vertical line at x_d .

4 Theoretical Framework for Network Learning

While supported as a research assistant by the grant, J. S. Judd developed a formal framework in which to address some important questions about the computational limits of network learning. He formalized a notion of learning in connectionist networks that characterizes the training of feed-forward networks. Considering different families of node functions, i.e., the functions that individual network elements compute, Judd proved that the learning problem, so formulated, is *NP*-complete and thus that it has no efficient general solution. One family of node functions studied is the set of logistic-linear functions, as used by the back-propagation algorithm. Additional theoretical results describe special classes of network topologies that can be trained in polynomial time.

Essentially, the learning goal as formulated is to find *one* algorithm that is *guaranteed* to load *any* performable task in *any* conceivable feed-forward network, where loading a task means specifying the functions implemented by all the nodes in the network. It was proved that this problem has no efficient general solution. However, several ways were considered to weaken the formulation so as to possibly yield an achievable goal. There may be large useful classes of networks (defined by some design restrictions) where loading a task would always be achievable in polynomial time. One can imagine several ways to constrain the class of networks and/or tasks and, or other aspects in such a way that the new loading problem would have some special regularity that might facilitate

its solution. A wide range of questions regarding narrowed or altered models of the connectionist learning goal are discussed in ref. [19]. Answers to some of these questions will assist connectionist learning research by narrowing its focus to those cases that hold the promise of scaling up. We believe that there is a great need for theory of this kind to increase the sophistication of connectionist research.

Several publications by Judd describe aspects of this research: refs. [20, 18, 17]. Judd received the Ph.D. degree in September, 1988, and a revised version of his dissertation on this subject will be published as a book by The MIT Press. Judd's work is receiving considerable attention—not only because it contains some of the few results of this kind about neural networks—but also because it may have practical significance for designing networks that are easy to train. Judd was appointed Adjunct Assistant Professor at the University of Massachusetts, and is now a Visiting Professor at the California Institute of Technology in Pasadena, where he is continuing this research.

5 Increasing Learning Rate through Learning Rate Adaptation

Despite the negative nature of Judd's results on the scaling up of network learning, we pursued several approaches to increasing learning rates of networks. The first approach is to alter, during the learning process, the parameters that determine how the size and direction of steps in weight space are computed as functions of the error gradient or gradient estimate. For example, the use of "momentum" in the error back-propagation method is an example of this approach. Although such methods cannot profoundly increase the size of networks that can be feasibly trained, they are nevertheless useful in practice for reducing the amount of computation required for studying network learning. The second approach we pursued is to develop networks whose architectures are well-suited for specific tasks. This approach, an example of which is discussed in Section 6, is not subject to Judd's theorem because it involves structuring network architectures and learning tasks instead of seeking a fast algorithm for arbitrary networks and tasks.

Methods such as Newton's method, recursive least squares, and conjugate-gradient methods can be regarded as including sophisticated means for adjusting learning rate parameters during learning. However, these methods cannot be implemented by neuron-like adaptive elements unless updating each weight uses information about the input signals on all of the unit's input pathways. That is, these methods are not as local as the simpler methods used in most connectionist research. Extending these methods to entire networks of units again requires the use of information that is not likely to be locally available to the units in real neural networks (or would be difficult to supply in VLSI implementations of artificial neural networks). We therefore focused attention on methods that conform to a locality constraint.

R. A. Jacobs, a graduate student supported by the grant, examined local methods of

adaptively adjusting learning rate parameters that have been proposed in the engineering literature, developed several new methods, and tested these methods in a selection of layered-network learning tasks. He summarized his findings in four heuristics that provide guidelines for how to achieve faster rates of convergence than steepest descent techniques: first, every parameter of the performance measure to be minimized should have its own individual learning rate; second, every learning rate should be allowed to vary over time; third, when the derivative of a parameter possesses the same sign for several consecutive time steps, the learning rate for that parameter should be increased; fourth, when the sign of the derivative of a parameter alternates for several consecutive time steps, the learning rate for that parameter should be decreased.

We studied one method for modifying rate parameters that we called the "delta-delta" rule. This rule performs steepest descent on an error surface defined over learning rate parameter space. If ϵ_i is the learning rate for the i^{th} weight, then it is updated according to:

$$\Delta \epsilon_i(t) = \gamma \frac{\partial J(t)}{\partial w_i(t)} \frac{\partial J(t-1)}{\partial w_i(t-1)},$$

where J is the function of the weights that is being minimized by the network and γ is a parameter. This algorithm for updating the learning rates implements the heuristics listed above. When the sign of the derivative of a weight is the same on consecutive time steps, the algorithm increases the learning rate for that weight. When the sign of the derivative of a weight alternates on consecutive time steps, the algorithm decreases the learning rate for that weight. Unfortunately, there are several problems with this rule that limit its practical use. To remedy these difficulties, we developed a related algorithm called the "delta-bar-delta" rule that is a bit more complicated in that it uses an exponential average of the current and past partial derivatives.

We compared the performance of several rules for updating learning rate parameters by applying them to four tasks. These tasks were the optimization of quadratic surfaces, and the learning of the exclusive-or, multiplexer, and binary-to-local functions. These tasks were chosen because their error surfaces possess a variety of terrains. The update rules tested were: steepest descent, momentum, delta-bar-delta, and a hybrid algorithm that combines the momentum and delta-bar-delta procedures. The last three tasks require the use of multi-layer networks. For all algorithms, the back-propagation procedure was used to calculate the partial derivative of the error with respect to each weight. The simulation results provide support for the four heuristics for how to achieve rates of convergence substantially faster than steepest descent algorithms and show that the delta-bar-delta and hybrid methods substantially accelerate learning. A paper by Jacobs describing these results were published in the October 1988 issue of *Neural Networks* [12].

Although the approach to accelerating learning using procedures of this kind does yield speed increases, it is not likely that this approach will make a significant difference for large problems (especially in light of Judd's theorem). We therefore began to focus on methods for accelerating learning that we believe can have a greater impact in practical

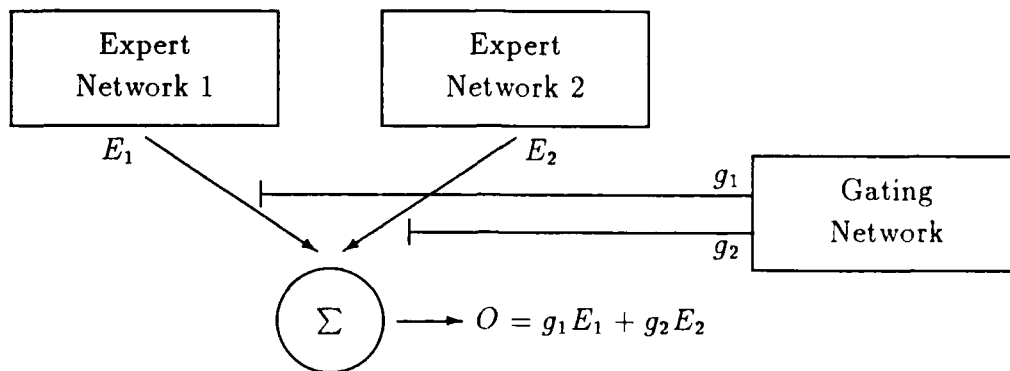


Figure 5: A modular network consisting of two expert networks and a gating network.

applications. These methods, described next, are based on structuring both the training process and the networks.

6 Modular Network Architectures

An approach to improving the learning ability of connectionist systems is to organize several networks into modular architectures. One advantage of such a structure is that individual networks are not faced with solving a large problem in its entirety. Large problems are solved by the combined efforts of several networks. This requires that a problem be broken into subproblems, and subproblems into subsubproblems, etc. We developed a learning method for a modular architecture consisting of several networks, which we call "expert networks", specialized for different kinds of tasks, and a "gating network" that learns how to switch in the best expert network for a particular subtask. The expert networks compete to learn about training patterns. Through such competition, different expert networks are allocated to learn different functions. This approach can accelerate the learning process if the architectures of the expert networks are designed based on some prior knowledge of subtasks, and it can permit the modular network to efficiently learn to perform multiple tasks by allocating different expert networks for each task.

Consider the architecture illustrated in Figure 5. It contains two types of networks. The expert networks compete to learn and perform training patterns. The gating network mediates this competition. After training, expert networks 1 and 2 compute different functions that are useful in different regions of the domain. Let the output of these networks be labeled E_1 and E_2 respectively. The gating network is an administrative agency that decides whether expert network 1 or 2 is currently applicable. This network

contains two output units labeled g_1 and g_2 respectively. The output of the system, O , equals $g_1 E_1 + g_2 E_2$. Therefore, when $g_1 = 1$ and $g_2 = 0$, expert network 1 determines the output of the system. Similarly, when $g_1 = 0$ and $g_2 = 1$, expert network 2 determines the output of the system.

During training, all networks modify their weights simultaneously using the back-propagation algorithm [30]. However, the expert and gating networks attempt to minimize different error functions. At each time step, the expert networks attempt to minimize the sum of squared error between the output of the system, O , and the desired output, O^* . This error function is written

$$J_O = \frac{1}{2}(O^* - O)^T(O^* - O). \quad (1)$$

The gating network attempts to minimize a more complicated error function. The intuition behind this function is as follows. For each training pattern, one expert network comes closer to producing the desired output than the other expert networks. In the competition among networks, this one is called the winner and all others are losers. Suppose that on this training pattern, the system's performance is significantly better than it has been in the past. In this case, the output of the gating network corresponding to the winning expert network is increased towards one and the outputs corresponding to the losing expert networks are decreased toward zero. Alternatively, if the system's performance has not improved, then all outputs of the gating network are moved toward a neutral value.

Mathematically, this intuition is expressed as follows. First, we determine if the system's performance is significantly better than it has been in the past. If t is the current time step, then the error $J_O(t)$ is a measure of the current performance. The measure of the system's past performance is the exponential average over time of J_O . This value, labeled $\overline{J_O}$, is computed by

$$\overline{J_O}(t) = \alpha J_O(t) + (1 - \alpha)\overline{J_O}(t - 1). \quad (2)$$

We use the binary variables λ_{WTA} (WTA stands for winner-take-all) and λ_{NT} (NT stands for neutral) to indicate whether the system's performance has significantly improved. Specifically,

$$\text{If } J_O(t) < \gamma \overline{J_O}(t - 1) \quad (3)$$

Then $\lambda_{WTA} = 1$ and $\lambda_{NT} = 0$

Else $\lambda_{WTA} = 0$ and $\lambda_{NT} = 1$.

Suppose that the system's performance has significantly improved. In this case, we determine which expert network's output is closest to the desired output. Define the error for expert network i to be the sum of squared error between the network's output E_i and the desired output O^* . This value, labeled J_{E_i} , is written

$$J_{E_i} = \frac{1}{2}(O^* - E_i)^T(O^* - E_i). \quad (4)$$

The winning expert network, labeled w , is the network with the smallest error. All other expert networks are losers. The desired value of the output of the gating network corresponding to the winning expert network, labeled g_w , is set to 1. The desired values of the outputs corresponding to the losing expert networks, labeled g_i , are set to 0. Alternatively, if the system's performance has not significantly improved, then the desired values for all outputs of the gating network are set to a neutral value. This value is $\frac{1}{n}$, where n is the number of expert networks.

The gating network's error function is:

$$\begin{aligned}
 J_G = & \lambda_{wTA} \frac{1}{2} \sum_{i=1}^n (g_i^* - g_i)^2 + \\
 & \lambda_{wTA} \frac{1}{2} (1 - \sum_{i=1}^n g_i)^2 + \\
 & \lambda_{wTA} \sum_{i=1}^n g_i (1 - g_i) + \\
 & \lambda_{NT} \frac{1}{2} \sum_{i=1}^n (\frac{1}{n} - g_i)^2.
 \end{aligned} \tag{5}$$

Only the first three terms contribute to the error when the system's performance has significantly improved. Otherwise, only the fourth term contributes to the error. The first term is the sum of squared error between the desired outputs and the actual outputs of the gating network. Minimization of the second term occurs when the outputs of the gating network sum to one. Minimization of the third term occurs when the outputs of the gating network are binary valued. The effect of minimizing the second and third terms is that, in response to each input pattern, one output of the gating network equals one and all others equal zero. The fourth term is the sum of squared error between the neutral value and the actual outputs of the gating network.

The gating network determines how much each expert network learns about each training pattern. Referring to Figure 5, note that the error vector back-propagated into expert network 1 is $g_1(O^* - O)$ and the error vector back-propagated into expert network 2 is $g_2(O^* - O)$. Thus, the gating network determines the magnitudes of the expert networks' error vectors.

Several investigators have noted that the selection of a network's topology is extremely important since the topology determines what functions the network can readily learn and what functions it can only learn with great difficulty, if at all. Furthermore, the topology also influences a network's ability to generalize. Frequently, an experimenter can use domain knowledge to select a set of expert network topologies that are potentially useful for rapidly learning the tasks faced by the architecture. An advantage of requiring the expert networks to compete to learn and perform training patterns is that the network whose topology most facilitates the learning of the function that generates the current training patterns is likely to win the competition. Thus, our architecture tends to allocate to each function an expert network with a topology that is appropriate to that function.

We tested this architecture on a simple vision task and on a robotics task. The vision task was proposed by Rueckl, Cave, and Kosslyn [29] who compared the performance of two connectionist systems on an object recognition task (henceforth, referred to as the "what" task) and a spatial localization task (henceforth, referred to as the "where" task). In their study, the first system is a single network which is required to learn both tasks. The second system, on the other hand, consists of two networks, one for each task. During training of the systems, one of nine patterns was placed at one of nine locations on a 5×5 matrix. The "what" task is to identify the pattern. The "where" task is to identify the spatial location. Rueckl, Cave, and Kosslyn [29] report that the second system is superior to the first system in the sense that it learns the tasks faster and develops a more interpretable representation.

An issue that Rueckl, Cave, and Kosslyn did not address, and the issue with which we were primarily concerned, is the development of a system that can *learn* if it is better to perform two or more tasks in distinct networks and, if so, can itself allocate distinct networks to learn each task. Such a system would have the ability to learn how to partition a task into subtasks and allocate these tasks to expert networks. Simulation results demonstrate that the architecture and learning rule we developed learns to allocate distinct expert networks to the "what" and "where" tasks. Furthermore, the architecture tends to allocate a single-layer network to the "where" task (this task is linearly separable) and a multi-layer network to the "what" task (this task is not linearly separable). Thus, these results suggest that the architecture learns to allocate to each task an expert network with a topology that is appropriate to that task.

The robotics task on which we tested this modular architecture is the task of learning to control a robot arm to move a variety of payloads, each with a different mass, along a desired trajectory. The architecture was successfully trained to serve as a feedforward controller for the robot arm using a training technique previously used by Kawato, Furukawa, and Suzuki [21] and Miller [26]. During training, the architecture learns to allocate one expert network to control the arm with no payload, a second expert network to control the arm with a light payload, and a third expert network to control the arm with a heavy payload.

We also trained a modification of the modular architecture to perform this trajectory-following task. This modified architecture includes a "share network" whose output contributes to the output of the system at all times. During training, the share network learns to control the arm with no payload and the expert networks learn to supply extra torques in order to compensate for the mass of each payload. In this sense, the modified architecture learns to solve a task by learning a shared strategy that is used in all contexts along with a set of modifications to this strategy that are applied in a context sensitive manner.

A preliminary discussion of this approach to modular architectures appeared as ref. [13], and Jacobs is currently writing a Ph.D. dissertation on this topic which we expect to be completed in the Spring of 1990.

7 Reinforcement Learning for Control of Dynamical Systems

In research funded by previous AFOSR grants, we applied adaptive networks to the "pole-balancing" task in which the network was required to learn how to prevent a pole from falling by exerting appropriate control actions [8, 32, 1]. Although we learned a lot from that research, its character was too heuristic to appeal directly to the adaptive control engineering community. We began the development of a more rigorous view of the type of method implemented by the pole-balancing controller. This effort has resulted in major insights into relationships between reinforcement learning methods for control and more orthodox engineering methods and, consequently, better understanding of what may be the strengths and weaknesses of connectionist reinforcement learning. In particular, it has become clear that the most relevant existing mathematical framework is the theory stochastic sequential decision problems and the most relevant computational methods are those of stochastic dynamic programming. We have studied these connections through interaction with C. Watkins, Philips Research Laboratories, whose Ph.D. dissertation [35] develops this connection, discussions with P. J. Werbos, of The National Science Foundation, who began exploring these connections in the mid-1970s [36, 37], as well as continuing interaction with R. S. Sutton, of GTE Laboratories, Inc. These connections, which are briefly outlined here, are discussed in detail by Barto, Sutton, and Watkins [9]. There is a huge literature on sequential decision problems and dynamic programming. A relatively recent and concise account is provided by Ross [28].

Stochastic sequential decision problems involve a decision-making system (let us call it the Decision Maker, or DM) interacting with a dynamical system in such a way that at the beginning of each of a series of discrete time periods, the DM observes the system's current state. Based on the observed state, the DM selects an action that will influence the system's behavior. After the action is performed, the DM receives a certain amount of *payoff* that depends on both the current system state and the action, and the system undergoes a state transition determined by its current state, the action that was performed, and random disturbances. Upon observing the new state, the DM chooses another action and continues in this manner for a sequence of time periods. The task of the DM is to form a rule for selecting actions, called a *policy*, that maximizes the expected value of the sum of the payoff earned over future time periods. The *return* of a policy refers to the sum of payoff received over time by a DM using that policy. The objective is therefore to form a policy that maximizes the expected return. These tasks are specific types of discrete-time control tasks where the policy corresponds to a state-feedback control law.

The number of time periods, each corresponding to the selection and performance of a single action, over which the return of a policy is determined is the *horizon* of the decision problem. It is usual to distinguish problems according to whether the horizon is finite or infinite. In finite-horizon problems, one desires a policy that maximizes the expected return over a given finite number of time periods. In infinite-horizon problems, one desires a policy that would maximize the expected return over an infinite number of

time periods. A *discount factor* is often used to weight payoff values so that the farther in the future a payoff is expected to occur, the less it contributes to the sum that is to be maximized. In this case, a policy's return is a weighted sum of the payoff values that the policy will produce over future time periods, where each weight depends both on the discount factor and when the payoff is received. The infinite-horizon discounted case is particularly interesting from a mathematical point of view and is the case to which our reinforcement learning are most closely related.

The return expected over the future depends on the discount factor, the current state of the system, and the policy the DM will use over the future. The *evaluation function* for a given policy and discount factor assigns to each state the expected discounted return given that the decision problem begins in that state and the DM uses the given policy over the entire future. The objective of the decision task is to find a policy (there may be many) such that, for a given discount factor, the corresponding evaluation function takes on values that are as large as possible. Such a policy is an *optimal policy*, and the evaluation function corresponding to it is the *optimal evaluation function*, which is unique for a given discount factor.

Because so many problems of practical interest can be formulated as stochastic sequential decision problems, there is an extensive literature devoted to the study of solution methods for this type of problem, the large majority of which require the decision maker to have a complete model of the dynamical system underlying the decision problem. Aside from extreme brute-force search methods, dynamic programming (DP) methods provide the only methods for solving these problems in the *general case of nonlinear systems*. Stochastic dynamic programming methods apply to stochastic sequential decision problems described above.

For finite-horizon problems, DP techniques work by computing backwards from the end of a problem to its beginning, calculating information pertinent to decision making at each step based on information previously calculated from that step to the problem's end. In the stochastic case, if there is one step remaining in the task, the expected return for each possible action can be computed on the basis of the knowledge—assumed to be available—about the system state transitions and payoff probabilities. Thus, for each state-action pair, one computes the payoff expected in one step, i.e., the expected one-step return. Any optimal decision policy must select the action that maximizes this expected one-step return when there is one step remaining in the decision problem. Then, given that we know the maximal expected return from each state for a one-step problem (which we have just computed), we can compute the expected two-step return for each state-action pair by treating the two-step problem as a one-step problem where the expected return is the expected immediate payoff on the first step plus the expected return for one more step—which is the quantity already computed. The optimal decision for the penultimate step of the problem selects the action that maximizes this expected two-step return. This process repeats until the entire optimal decision policy is specified. If the problem has an infinite horizon, this iterative method can be modified slightly so that it successively approximates the infinite-horizon case.

Stochastic DP requires scales very poorly to large problems. As the number of process states, actions, and steps in the decision problem increase, the amount of computation required quickly becomes prohibitive. Consequently, the problem of forming *estimates* of optimal evaluation functions and optimal policies without performing all of this computation has great practical significance. The reinforcement learning methods that we have studied can be seen as such approximation methods that have the additional property of being applicable when complete knowledge of the dynamical system underlying a decision task is absent.

When a complete model of the dynamical system underlying a sequential decision task is not available, it is necessary to learn about the system while interacting with it. One approach is to construct a model of the system underlying the decision problem in the form of estimates of state-transition and payoff probabilities and then apply DP methods under the assumption that the system model is accurate (e.g., refs. [24, 25, 31]). In the nonlinear case, these methods scale very poorly because they require repeated application of DP methods. Another approach is to directly adjust the decision policy as a result of observed consequences of the decisions it specifies. Here, the DM tries out a variety of decisions, observes their consequences, and directly adjusts its policy in order to improve it. It is possible to facilitate this direct learning of a decision policy by combining it with a process for estimating an evaluation function so that the long-term consequences of actions are reflected in evaluations that are available immediately after an action is performed. This is the approach we took in the pole-balancing system [8], where the "Associative Search Element" adjusted the policy and the "Adaptive Critic Element" estimated the evaluation function corresponding to the evolving policy. In fact, as discussed in ref. [9], the learning rule used by the Adaptive Critic Element can be understood in terms of a functional equation from DP. Although this is a "model-free" approach to learning a decision policy, it does not preclude the additional use of system models. Methods combining model-free techniques with model-based methods will be a major emphasis of future research.

Within the framework of sequential decision problems and DP methods, the connectionist reinforcement learning methods we have studied are best viewed as Monte Carlo methods for approximating the results of stochastic DP methods and are applicable when there is no complete model of the dynamical system underlying the decision task. Instead of computing optimal policies and evaluation functions using a system model and DP methods, these functions are directly approximated by connectionist networks on the basis of sequences of trials with the decision task. Understanding these relationships to existing theories has greatly contributed to our goal of establishing connectionist reinforcement learning methods as rigorously defensible approaches to learning control applicable to complex nonlinear control tasks. As pointed out by Werbos [37], coupling connectionist function approximation techniques to Monte Carlo DP provides a means for bringing connectionist algorithms and hardware to bear on sequential decision tasks that are too large and involve too much uncertainty to permit solution by existing exact methods.

8 Imperfect State Information

Methods for solving sequential decision problems based on dynamic programming such as those discussed in Section 7 rely on having access to the state of the dynamical system underlying the decision problem. When this information is absent, additional methods must be used to provide estimates of the current state of the system. Additional complexities arise if the problem is not just to estimate the state of a known dynamical system but to construct a dynamical model based on observable information whose states are to provide input to the decision-making process. Although there exists a large body of literature on state estimation and system identification, we have pursued some ideas that are not easily placed within the spectrum of traditional methods, having stronger ties to grammatical inference than to traditional engineering methods. Below is a brief description of this research which makes up part of the Ph.D. research of J. Bachrach, one of the graduate students supported by the grant, who is currently writing a dissertation on this topic expected to be completed in the Spring or Summer of 1990.

This work began with an investigation of methods for training simple reverberatory circuits to act as memory devices. For example, a connectionist unit that excites itself through a recurrent connection can be "set" or "reset" like an SR flipflop. This kind of memory is different from the kind of long-term memory that is stored in connection weights. The problem is to learn *when* to set or reset these bits in a variety of paradigms. This kind of knowledge *would* be stored in connection weights. Early approaches to this type of problem led to an investigation of research being conducted by Schapire and Rivest of the MIT Laboratory for Computer Science, who designed an algorithm for constructing a model of a finite-state environment through exploration [27]. We have developed a connectionist network based on the representation of finite-state automata used in the Rivest-Schapire (RS) algorithm. This representation has a natural, direct connectionist implementation and is able to strongly constrain the network's architecture. Although the network explores the environment in the simplest possible way—by choosing random actions—for simple environments, the network can outperform the Rivest and Schapire algorithm because it is able to consider many hypotheses in parallel. The network has the additional strength that it is applicable to nondeterministic environments.

As a simple example, consider an environment consisting of n rooms arranged in a circle, with a light and light switch in each room. In a given room, the decision maker can take one of three actions: move to the room on the left, move to the room on the right, and toggle the light switch in the current room. The decision maker can sense the state of the light in the current room (*on* or *off*). This environment can be modeled in the obvious way by a finite-state automaton (FSA) having 2^n states. Although one could try learning an unstructured representation of this FSA by estimating its state transition function, it often is not efficient to do so because the unstructured FSA representation does not capture redundancy inherent in the environment. For example, in this n -room environment, although the sensation resulting from toggling the light switch is dependent on only the state of the current room, in the unstructured FSA representation, knowledge

about "toggle" must be encoded for each of the 2^n distinct states. The environment has symmetries not represented in the unstructured FSA representation. Rather than trying to learn the FSA in unstructured form, Rivest and Schapire suggest learning another representation called an *update graph*. The advantage of the update graph representation is that in environments with many regularities, the number of nodes in the update graph can be much less than the number of states of the FSA (e.g., $2n$ versus 2^n for the n -room world). The update graph is a particular structured representation of the FSA in which each state is represented by a pattern of activity across the nodes of the graph. In other words, the update graph provides a particular distributed representation of environmental states.

The update graph representation is based on the notion of a *test*. A test consists of a sequence of zero or more actions followed by the application of a predicate that is true for a particular sensation. A test is performed by executing the sequence of actions from the current environmental state and then checking for the presence or absence of the sensation. Certain tests will always yield the same truth value independently of the current environmental state. For example, toggling the light switch four times has exactly the same effect as toggling the switch two or zero times. Such tests are *equivalent*, and there is a node in the update graph representation for each equivalent class of tests. Each directed arc of the update graph is labeled with an action. There is an arc directed from node α to node β labeled *action* if the test resulting from executing any test in the equivalence class β followed by executing *action* is in the equivalence class represented by node α . Associated with each node is a binary variable giving the truth value of the corresponding test given the current environmental state. If the current values of all nodes are known, then the values after executing an action can be inferred from the update graph: The value of node β following *action* is equal to the current value of the node α connected to β with the link labeled *action*. Thus, the sensations obtained after performing a sequence of actions can be predicted simply by shifting values around in the update graph. The update graph serves as a structured model of the environment.

Following this work of Schapire and Rivest [27], Bachrach in collaboration with M. Mozer of the University of Colorado, devised a network architecture that learns to perform as an update graph (Figure 6). Each unit in the network corresponds to a node in the update graph. The binary-valued activity of a unit corresponds to the truth value of a node. Connections between units are gated by a set of *gating units* such that the connection is enabled only if the given action is performed by the organism; this corresponds to the labeled links between nodes of the update graph.

Training networks of the form shown in Figure 6 to represent update graphs relies on performing error back-propagation through time [30] while the the decision maker is interacting with the environment using a random policy. In simple environments, the connectionist update graph outperforms the RS algorithm even though the action sequence used to train the network is generated at random, whereas the RS algorithm uses a specific exploration strategy. We conjecture that the network does as well as it does because it considers and updates many hypotheses in parallel at each time step.

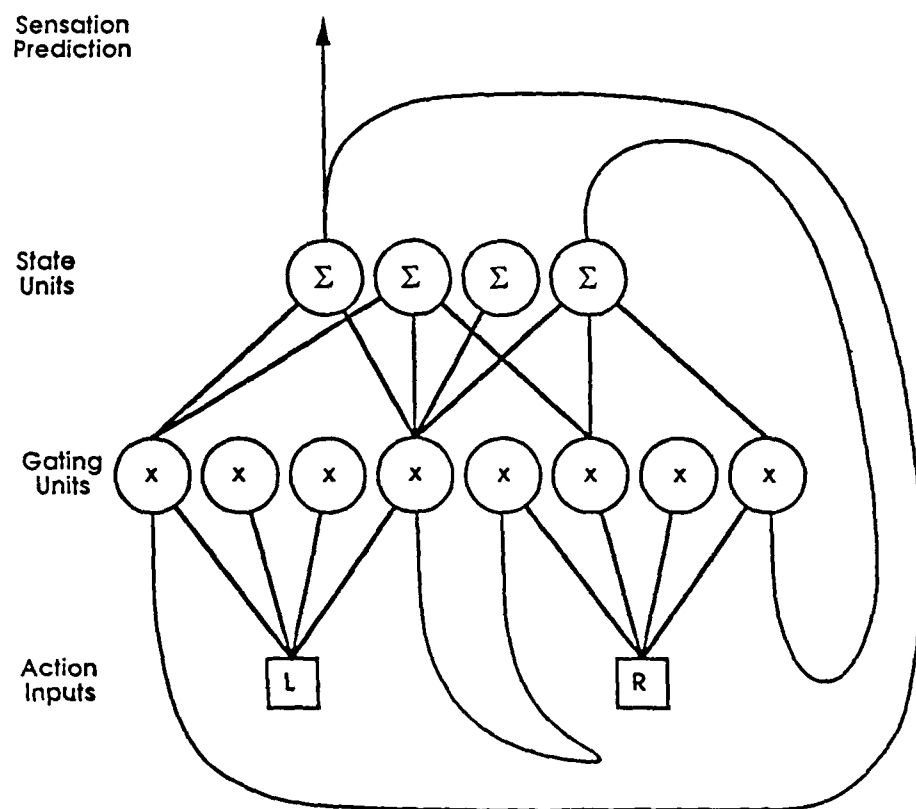


Figure 6: Network architecture for learning update-graph representations of finite-state environments.

The network is also able to construct models of stochastic environments. For example, if the sensations in the 3-room world are slightly unreliable, the network still learns the task. The RS algorithm cannot handle nondeterminism. In more complex environments, however, the network does not perform as well. For example, it failed to learn a 32-room environment, whereas the RS algorithm succeeded. An intelligent exploration strategy seems necessary in this case.

In order to further develop these and other ideas, a test-bed was designed and implemented for studying them as applied to spatial navigation problems. This test-bed is described in the next section.

9 Spatial Navigation Test-Bed

A wide variety of sequential decision tasks can be formulated in terms of moving in spatial environments while receiving sensory information providing clues as to location and orientation. Some of our initial explorations of reinforcement learning networks were conducted in this domain [7, 5], and we have continued to find this a good domain for posing problems and investigating solution methods (a recent example is described in ref. [9]). Following is a brief description of a test-bed we implemented that will allow us to address basic learning issues while at the same time provide us with fairly realistic simulations of robot navigation tasks.

The system simulates a cylindrical robot with four wheels and a 360° sensor belt. The simulated robot can translate independently and simultaneously in both the x and y directions relative to its orientation. The motion of the robot is simulated as discrete movements, one per time step. The robot has 16 distance sensors and 16 grey-scale sensors evenly placed around its perimeter. The distance sensors roughly simulate sonar. Information from these simulated sensors is processed to yield 16 distance values and 16 grey-scale sensor values which measure the intensity of light at the various orientations.

Figure 7 shows a display created by the navigation simulator. The bottom portion of the figure shows the robot's environment as seen from above. In this display, the bold circle represents the robot's "home" position, with the radius line indicating the home orientation for a homing task. The other circle with radius line represents the robot's current position and orientation. The topmost canvas shows the grey-scale view from the home position and orientation, and the next canvas shows the grey-scale view from the robot's current position and orientation. The third canvas from the top shows smoothed distance values from both positions and orientations, with those from home shown in horizontal stripes and those from current shown with vertical stripes. The fourth canvas shows smoothed grey-scale images for both the home and current positions and orientations. The fifth panel from the top of the figure shows the actions of the robot, from left to right, x , y , and rotation.

This test-bed will be used by Bachrach for extending the approach to constructing

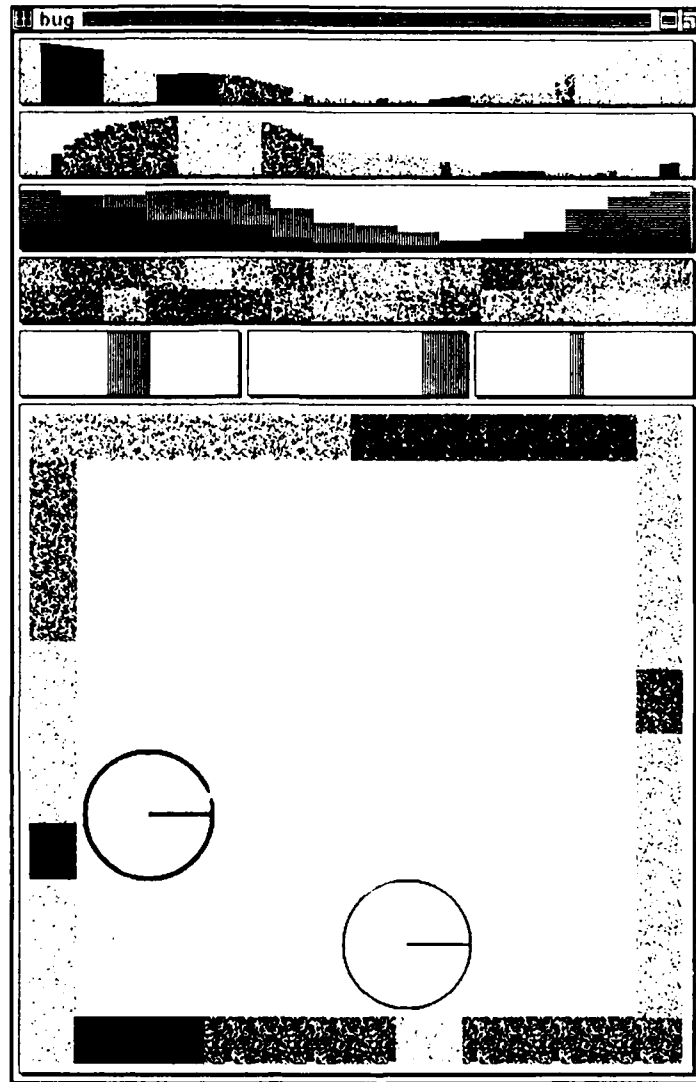


Figure 7: Computer display generated by the navigation simulator.

environmental models described above in Section 8. He will study homing tasks in which several positions and orientations produce the same sensory stimulation.

10 Conclusion

Progress was made in the development of connectionist learning methods permitting networks to learn when they cannot be provided with training information of the high quality required by supervised-learning methods. These methods can permit the application of adaptive connectionist networks to a tasks involving complex dynamical behavior and high degrees of uncertainty. The various projects undertaken with the support of this grant were all motivated by issues related to the control by networks of dynamical systems. It is apparent that connectionist techniques can substantively contribute to the theory and practice of control of nonlinear dynamical systems with many degrees-of-freedom. Future research will be directed toward studying these methods as applied to a variety of simulated and real control tasks.

11 Publications, Interactions, and Advanced Degrees

11.1 Publications

A. G. Barto and M. I. Jordan, "Gradient following without back-propagation in layered networks," in *Proceedings of the IEEE First Annual Conference on Neural Networks*, vol. 2, pp. 629-636, San Diego, CA, 1987.

S. Judd, "Learning in networks is hard," in *Proceedings of the IEEE First Annual Conference on Neural Networks*, Vol. 2, pp. 685-692, San Diego, CA, 1987.

R. S. Sutton and A. G. Barto, "A temporal-difference model of classical conditioning," in *Program of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, Washington, July 1987.

S. Judd, "Complexity of learning with various node functions," COINS Technical Report 87-60, University of Massachusetts, Amherst, MA.

M. I. Jordan and D. A. Rosenbaum, "Action," in M. I. Posner (Ed.) *Foundations of Cognitive Science*, MIT Press, Cambridge MA, 1989 (also COINS Technical Report 88-26, March 1988, Computer and Information Science, University of Massachusetts, Amherst MA).

M. I. Jordan, "Supervised learning and systems with excess degrees of freedom," COINS Technical Report 88-27, May 1988, Computer and Information Science, University of Massachusetts, Amherst MA.

A. G. Barto, "From chemotaxis to cooperativity: Abstract exercises in neuronal learning strategies," in *The Computing Neuron*, R. Durbin, R. Maill, and G. Mitchison (Eds.), Addison-Wesley (also COINS Technical Report 88-65, 1988, Computer and Information Science, University of Massachusetts, Amherst MA).

R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, Vol. 1, pp. 295-307, 1988.

R. A. Jacobs, "Initial experiments on constructing domains of expertise and hierarchies in connectionist systems," in *Proceedings of the 1988 Connectionist Models Summer School*, pp. 144-153, Morgan-Kaufmann, 1988.

V. Gullapalli, "A stochastic algorithm for learning real-valued functions via reinforcement feedback," COINS Technical Report 88-91, 1988, Computer and Information Science, University of Massachusetts, Amherst (submitted to *Neural Networks*).

J. S. Judd, "On the complexity of loading shallow networks," *Journal of Complexity*, Special Issue on Neural Computation, September, 1988.

J. S. Judd, "Neural network design and the complexity of learning." Ph.D. dissertation, Department of Computer and Information Science, University of Massachusetts, September, 1988. A revised version of this dissertation will be published in book form by The

MIT Press.

R. S. Sutton and A. G. Barto, "A time-derivative theory of Pavlovian conditioning," to appear in *Learning and Computational Neuroscience*, M. Gabriel and J.W. Moore (Eds.), Cambridge, MA: The MIT Press.

"Connectionist learning for control: An overview," to appear in *Neural Networks for Control*, T. Miller, R.S. Sutton, and P.J. Werbos (Eds.), Cambridge, MA: The MIT Press (also COINS Technical Report 89-89, 1989, Computer and Information Science, University of Massachusetts, Amherst MA).

A. G. Barto, R. S. Sutton, and C. W. C. H. Watkins, "Learning and sequential decision problems," to appear in *Learning and Computational Neuroscience*, M. Gabriel and J.W. Moore (Eds.), Cambridge, MA: The MIT Press (also COINS Technical Report 89-95, 1989, Computer and Information Science, University of Massachusetts, Amherst MA)

11.2 Interactions

M. I. Jordan, seminar at the Department of Psychology, University of Oregon, November, 1986.

A. G. Barto, Texas Instruments Research Colloquium, March 6, 1987, Dallas TX. Contact persons: Andrew Penz, M. Gately.

A. G. Barto, "Connectionist learning control," presentation at the Conference on Neural Networks For Computing, Snowbird, Utah, April 1-4, 1987.

M. I. Jordan, "Domain models and systems with excess degrees of freedom," presentation at the Conference on Neural Networks For Computing, Snowbird, Utah, April 1-4, 1987.

A. G. Barto, Computer Science and Engineering Colloquium, Oregon Graduate Center, Beaverton OR, April 7, 1987. Contact person: D. Hammerstrom.

M. I. Jordan, seminar at the Department of Psychology, University of California at San Diego, April, 1987.

M. I. Jordan, seminar at the Department of Brain and Cognitive Science, MIT, March, 1987.

A. G. Barto, "Game-theoretic cooperativity in networks of self-interested units." invited presentation at University of Maryland's Institute for Advanced Computer Studies Workshop on Connectionist Models in Computational and Cognitive Science, University of Maryland, May 3-5, 1987.

M. I. Jordan, seminar at Bell Communications Research, 435 South St., Morristown, NJ, May, 1987. Contact person: J. Alspector.

A. G. Barto, seminar at Bell Communications Research, 435 South St., Morristown, NJ,

May, 1987. Contact person: J. Alspector.

A. G. Barto, Applied and Computational Mathematics Seminar at BBN Corp., Cambridge, MA, May 1987. Contact person: A. Boulanger.

M. I. Jordan, seminar at the Department of Speech and Communication, MIT, May, 1987.

A. G. Barto, "Connectionist motor control: Coordination and adaptation," invited presentation at U.S.-Japan Joint Seminar: Competition and Cooperation in Neural Nets. Univ. of Southern California, LA, CA, May 18-22, 1987. Contact person: M. Arbib.

A. G. Barto, seminar at the Center for Biological Information Processing, Whitaker College, MIT: May 27, 1987. Contact person: E. Saund.

A. G. Barto and M. I. Jordan, "Gradient following without back-propagation in layered networks," presentation at the IEEE First Annual Conference on Neural Networks, San Diego, CA, 1987.

S. Judd, "Learning in networks is hard," presentation at the IEEE First Annual Conference on Neural Networks, San Diego, CA, 1987.

A. G. Barto, Department of Psychology Colloquium, University of Colorado, June 1987. Contact person: B. McNaughton.

A. G. Barto, seminar at Department of Physiology, Northwestern University, Chicago, ILL, October, 1987. Contact person: J. C. Houk.

A. G. Barto, seminar at King's College Seminar, King's College, Cambridge, UK, November 1987. Contact person: G. J. Mitchison.

A. G. Barto, presentation at the Air Force Office of Scientific Research Review of Air Force Sponsored Basic Research in Neuroscience, United States Air Force School of Aerospace Medicine, Brooks Air Force Base, Texas, November 1987.

A. G. Barto, presentation at workshop: Neurons, Nodes and Networks, University of Arizona Center for the Study of Complex Systems, Tucson, Arizona, March 1988. Contact person: L. Nadel.

A. G. Barto, Computer Science Ad Hoc Seminar, Microelectronics Center of North Carolina Communications Network Seminar Series, April, 1988. Contact person: Z. Ras.

A. G. Barto, seminar at Case Western Reserve University, Cleveland, April 1988. Contact person: Y. Pao.

A. G. Barto, presentation at workshop: The Neurone as a Computational Unit, Cambridge, UK, June 1988. Contact person: G. J. Mitchison.

A. G. Barto, Faculty member at the 1988 Connectionist Models Summer School, Carnegie-Mellon University, June 1988. Contact person: D. Touretzky.

A. G. Barto, presentation at the NSF Workshop on the Applications of Neural Networks to Robotics and Control, The University of New Hampshire, October 16-18, 1988. Contact persons: T. Miller, R. Sutton, and P. Werbos.

A. G. Barto, seminar at Oak Ridge National Laboratories, Oak Ridge, Tennessee, November, 1988. Contact person: G. Liepins.

A. G. Barto, seminar at the Computer Science Department, University of Tennessee, November, 1988. Contact person: G. Liepins.

A. G. Barto, seminar at the Center for Adaptive Systems, Boston University, Boston, MA, December, 1988. Contact person: S. Grossberg.

A. G. Barto, seminar at the Advanced Telecommunications Research Institute International (ATR) Auditory and Visual Perception Research Laboratories, Osaka, Japan, January, 1989. Contact person: M. Kawato.

A. G. Barto, presentation at The Rank Prize Funds Mini-Symposium on Neural Network Computation, Broadway, Worcestershire, England, February, 1989. Contact persons: H. Barlow (Cambridge) and M. Brady (Oxford).

M. Mozer and J. Bachrach, "Discovering the structure of a reactive environment by exploration," presentation at the Conference on Neural Networks for Computing, Snowbird, Utah, 1989.

A. G. Barto, seminar at the School of Engineering and Applied Science, University of Durham, Durham, England, May, 1989. Contact person: P. Mars.

A. G. Barto, seminar at Edinburgh University, Edinburgh, Scotland, May, 1989. Contact person: D. Willshaw.

A. G. Barto, seminar at the Department of Psychology, Cambridge University, Cambridge, England, May, 1989. Contact person: A. Dickinson.

A. G. Barto, seminar at the Department of Computer Science, University of Oxford, Oxford, England, May, 1989. Contact person: M. Brady.

A. G. Barto, seminar at the Applied Psychology Unit, Cambridge, England, June, 1989. Contact person: D. Shanks.

A. G. Barto, presentation at the workshop: The Computational Principles of the Cerebral Cortex, King's College, Cambridge, England, July, 1989. Contact person: G. J. Mitchison.

A. G. Barto, four public lectures on connectionist modeling, University of New South Wales, Sydney, Australia, July 1-4, 1989. Contact person: E. J. Kehoe.

A. G. Barto, seminar at the Department of Computer Science, University of Sydney, Sydney, Australia, July, 1989. Contact person: N. Foo.

11.3 Participating Professionals

Following is a list of professionals who participated directly in research partially or completely funded by AFOSR-87-0030, or closely related research.

Dr. R. S. Sutton, GTE Laboratories Incorporated, Waltham, MA. Dr. Sutton, formerly a student of Barto whose Ph.D. research was supported by previous AFOSR grants, has continued to interact closely with Barto and students funded by AFOSR-87-0030. In the period being reported here, Sutton and Barto interacted in writing a conference paper (ref. [34]) and two book chapters (refs. [33, 9]). Dr. Sutton has served as a member of the Master's committees of several students funded by AFOSR-87-0030.

Consultation with Dr. P. S. Sastry, Indian Institute of Science, Bangalore, India, on stochastic convergence theory of A_{R-P} and related algorithms. Period: 5/11/88-6/19/88. Barto and Sastry are continuing to correspond regarding this topic.

Dr. M. I. Jordan, Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA. Dr. Jordan was supported as a Post-Doctoral Research Associate by AFOSR-87-0030 until he began his current position as Assistant Professor at MIT in January, 1988. Since that time, he has maintained active interaction with researchers funded by AFOSR-87-0030 and is serving on the Ph.D. committees of several of the graduate students supported by this grant.

Interaction with C. J. C. H. Watkins, Philips Research Laboratories, Cross Oak Lane, Redhill Surrey RH1 5HA, England. While employed at Philips, Watkins pursued a Ph.D. in Psychology at the University of Cambridge, Cambridge, England. His dissertation, completed in June, 1989, concerns the problems of learning with delayed rewards, a topic partially inspired by the AFOSR funded research of our group on reinforcement learning. Watkins elaborated connections between our approach and concepts and computational methods from the theory of stochastic dynamic programming. Watkins collaborated with Barto and Sutton in producing the technical report on this subject, ref. [9], due to appear as a book chapter.

Interaction with Dr. J. C. Houk, Chairman, Department of Physiology, Northwestern University Medical Center, Chicago, Illinois. Dr. Houk began a sabbatical semester as a Visiting Professor in the Department of Computer Science, University of Massachusetts, in Sept. 1988. He is Principal Investigator of ONR Grant N00014-88-K0339, which is supporting a computer science graduate student at the University of Massachusetts and which lists Barto as a consultant. This project is directed toward constructing a model of the cerebellum as a trainable pattern generator, and is closely related to the research supported by AFOSR reported here.

11.4 Advanced Degrees

Following is a list of advanced degrees awarded to students who were partially or completely supported by AFOSR-87-0030 while graduate students in the Department of Computer Science, University of Massachusetts.

J. S. Judd was awarded the Ph.D. Degree in Computer and Information Science in September, 1988, for research supported by AFOSR-87-0030 and a previous AFOSR grant. His dissertation is entitled "Neural Network Design and the Complexity of Learning." A revised version of the dissertation will be published in book form by the MIT Press. Dr. Judd is currently an Adjunct Assistant Professor of Computer and Information Science, University of Massachusetts, and is a Visiting Professor at the California Institute of Technology, Pasadena, CA, where he is continuing this line of research.

R. A. Jacobs was awarded the M.S. Degree in May, 1987. His M.S. project was entitled "Increased Rates of Convergence Through Learning Rate Adaptation." A paper based on this project appeared in the journal *Neural Networks*. Jacobs is currently working on a Ph.D. dissertation on modular network architectures which is expected to be complete in the Spring of 1990.

V. Gullapalli was awarded the M.S. Degree in May, 1988. His M.S. project was entitled "Stochastic Reinforcement Learning in Motor Control" A paper based on this project is currently in review for the journal *Neural Networks*. Gullapalli is currently working on a Ph.D. dissertation on applying reinforcement learning to motor control problems which is expected to be complete in the Fall of 1990 or the Spring of 1991.

J. R. Bachrach was awarded the M.S. Degree in December, 1988. His M.S. project was entitled "Learning to Represent State." Bachrach is currently working on a Ph.D. dissertation on this same subject which is expected to be complete in the Spring or Summer of 1990.

V. Bauer completed an M.S. project with partial support from the grant in August, 1989. Her project was entitled "Effect of Discounting on Rate of Convergence in Temporal Difference Learning." Bauer will receive the M.S. degree in December, 1989, and will not pursue a Ph.D. degree at the present time.

References

- [1] C. W. Anderson. *Learning and Problem Solving with Multilayer Connectionist Systems*. PhD thesis, University of Massachusetts, Amherst, MA, 1986.
- [2] A. G. Barto. Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology*, 4:229-256, 1985.
- [3] A. G. Barto. From chemotaxis to cooperativity: Abstract exercises in neuronal learning strategies. In R. Durbin, R. Maill, and G. Mitchison, editors, *The Computing Neuron*. Addison-Wesley, 1989.
- [4] A. G. Barto and P. Anandan. Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:360-375, 1985.
- [5] A. G. Barto, C. W. Anderson, and R. S. Sutton. Synthesis of nonlinear control surfaces by a layered associative search network. *Biological Cybernetics*, 43:175-185, 1982.
- [6] A. G. Barto and M. I. Jordan. Gradient following without back-propagation in layered networks. In *Proceedings of the IEEE First Annual Conference on Neural Networks*, pages II629-II636, San Diego, CA, 1987.
- [7] A. G. Barto and R. S. Sutton. Landmark learning: An illustration of associative search. *Biological Cybernetics*, 42:1-8, 1981.
- [8] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835-846, 1983. Reprinted in J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*, The MIT Press, Cambridge, MA, 1988.
- [9] A. G. Barto, R. S. Sutton, and C. Watkins. Learning and sequential decision making. In M. Gabriel and J. W. Moore, editors, *Learning and Computational Neuroscience*. The MIT Press, Cambridge, MA. To appear.
- [10] V. Gullapalli. A stochastic algorithm for learning real-valued functions via reinforcement feedback. Technical Report 88-91, University of Massachusetts, Amherst, MA, 1988.
- [11] Y. C. Ho. Decision theory and information structures. *Proceedings of the IEEE*, 68:644-654, 1980.
- [12] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295-307, 1988.
- [13] R. A. Jacobs. Initial experiments on constructing domains of expertise and hierarchies in connectionist systems. In *Proceedings of the 1988 Connectionist Models Summer School*, 2929 Campus Drive, Suite 260, San Mateo, CA 94403, 1988. Morgan-Kaufman.
- [14] M. I. Jordan. *The Learning of Representations for Sequential Performance*. PhD thesis, University of California, San Diego, 1985.
- [15] M. I. Jordan. Supervised learning and systems with excess degrees of freedom. Technical Report 88-27, University of Massachusetts, Amherst, MA, 1988.

- [16] M. I. Jordan and D. A. Rosenbaum. Action. In M. I. Posner, editor, *Foundations of Cognitive Science*. The MIT Press, Cambridge, MA, 1989.
- [17] J. S. Judd. Complexity of connectionist learning with various node functions. Technical Report 87-60, University of Massachusetts, Amherst, MA, 1987.
- [18] J. S. Judd. Learning in networks is hard. In *Proceedings of the IEEE First Annual Conference on Neural Networks*, volume 2, pages 685-692, San Diego, CA, 1987.
- [19] J. S. Judd. *Neural Network Design and the Complexity of Learning*. PhD thesis, Computer and Information Science dept., University of Massachusetts, Amherst, Mass. U.S.A., 1988.
- [20] J. S. Judd. On the complexity of loading shallow neural networks. *Journal of Complexity*, September 1988. Special issue on Neural Computation, in press.
- [21] M. Kawato, K. Furukawa, and R. Suzuki. A hierarchical neural-network model for control and learning of voluntary movement. *Biological Cybernetics*, 57:169-185, 1987.
- [22] A. H. Klopff. Brain function and adaptive systems—A heterostatic theory. Technical Report AFCRL-72-0164, Air Force Cambridge Research Laboratories, Bedford, MA, 1972. A summary appears in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, 1974, IEEE Systems, Man, and Cybernetics Society, Dallas, TX.
- [23] A. H. Klopff. *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence*. Hemisphere, Washington, D.C., 1982.
- [24] P. R. Kumar and Woei Lin. Optimal adaptive controllers for unknown markov chains. *IEEE Transactions on Automatic Control*, 25, 1982.
- [25] P. Mandl. Estimation and control in markov chains. *Advances in Applied Probability*, 6:40-60, 1974.
- [26] W. T. Miller. Sensor based control of robotic manipulators using a general learning algorithm. *IEEE Journal of Robotics and Automation*, 3:157-165, 1987.
- [27] R. L. Rivest and R. E. Schapire. Diversity-based inference of finite automata. In *Proceedings of the Twenty-Eighth Annual Symposium on Foundations of Computer Science*, pages 78-87, 1987.
- [28] S. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, New York, 1983.
- [29] J. G. Rueckl, K. R. Cave, and S. M. Kosslyn. Why are 'what' and 'where' processed by separate cortical visual systems? a computational investigation. *Journal of Cognitive Neuroscience*, 1:171-186, 1989.
- [30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol.1: Foundations*. Bradford Books/MIT Press, Cambridge, MA, 1986.

- [31] M. Sato, K. Abe, and H. Takeda. Learning control of finite markov chains with unknown transition probabilities. *IEEE Transactions on Automatic Control*, 27, 1982.
- [32] R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, MA, 1984.
- [33] R. S. Sutton and A. G. Barto. Time-derivative models of pavlovian conditioning. In M. Gabriel and J. W. Moore, editors, *Learning and Computational Neuroscience*. The MIT Press, Cambridge, MA. To appear.
- [34] R. S. Sutton and A. G. Barto. A temporal-difference model of classical conditioning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA, 1987.
- [35] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989. Pending.
- [36] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.
- [37] P. J. Werbos. Generalization of back propagation with applications to a recurrent gas market model. *Neural Networks*, 1, 1988.
- [38] R. J. Williams. Reinforcement learning in connectionist networks: A mathematical analysis. Technical Report ICS 8605, Institute for Cognitive Science, University of California at San Diego, La Jolla, CA, 1986.
- [39] R. J. Williams. Reinforcement-learning connectionist systems. Technical Report NU-CCS-87-3, College of Computer Science, Northeastern University, 360 Huntington Avenue, Boston, MA, 1987.